```
/* Write a program to implement linear search using functions
Created on : 13-6-23
Author: Lasya Nalagandla, 22251A3656
*/
#include<stdio.h>
#define MAX 5
void search(int arr[],int key);
void main()
{
    int key,i,a[MAX];
    for(i=0;i<MAX;i++)
    {
        printf("enter element : ");
        scanf("%d",&a[i]);
    }
    printf("item that needs to be searched\n");
    scanf("%d",&key);
    search(a,key);
}
void search(int arr[],int key)
{
    for(int i=0;i<MAX;i++)
    {
        if(arr[i]==key)
        {
            printf("item found at position %d\n",i);
            return;
        }
    }
    printf("item not found\n");
}
/*output
student@student:~/22251A3656$ gcc ls.c
student@student:~/22251A3656$ ./a.out
enter element : 1
enter element : 2
enter element : 3
enter element : 4
enter element : 5
item that needs to be searched 7
item not found
student@student:~/22251A3656$ gcc ls.c
student@student:~/22251A3656$ ./a.out
enter element : 2
enter element : 3
enter element : 4
enter element : 6
enter element : 8
item that needs to be searched 8
item found at position 5 */
```

```
/*Write a program to implement binary search using functions
Created on : 13-6-23
Author: Lasya Nalagandla, 22251A3656
*/
#include<stdio.h>
#define MAX 5
void search(int a[],int key);
void main()
{
    int key,j,temp,i,a[MAX];
    for(i=0;i<MAX;i++)
    {
        printf("enter element : ");
        scanf("%d",&a[i]);
    }
    printf("item that needs to be searched\n");
    scanf("%d",&key);
    for(i=0;i<MAX;i++)
    {
        for(j=0;j<MAX-1;j++)
        {
            if(a[j]>a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
        }
    }
    printf("After sorting the elements are \n");
    for (i=0;i<MAX;i++)
    printf("a[%d]=%d\n",i,a[i]);
    search(a,key);
}
void search(int a[],int key)
{
    int low,high,mid,flag;
    low=0,high=MAX-1;
    while(low<=high)
    {
        mid=(low+high)/2;
        if(a[mid]==key)
        {
            flag=1;
            break;
        }
        else if(a[mid]<key)
        {
            low=mid+1;
        }
        else if(a[mid]>key)
        {
            high=mid-1;
        }
    }
    if(flag==1)
    printf("%d found at position %d\n",key,mid);
    else
    printf("%d not found\n",key);
}
/*output
student@student:~/22251A3656$ gcc bs.c
student@student:~/22251A3656$ ./a.out
enter element : 1
enter element : 2
enter element : 3
enter element : 4
enter element : 5
item that needs to be searched
4
After sorting the elements are
a[0]=1
a[1]=2
a[2]=3
a[3]=4
a[4]=5
4 found at position 3
student@student:~/22251A3656$ gcc bs.c
student@student:~/22251A3656$ ./a.out
enter element : 2
enter element : 4
enter element : 6
enter element : 8
enter element : 9
item that needs to be searched
1
After sorting the elements are
a[0]=2
a[1]=4
a[2]=6
a[3]=8
a[4]=9
1 not found
*/
```

```
/*Write a program to implement linear search using recursion
Created on : 13-6-23
Author: Lasya Nalagandla, 22251A3656
*/

#include<stdio.h>
int search(int a[],int,int);
int key,i,front=0,found,n,a[50];
void main()
{
    printf("enter the number of elements u want to read\n");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("enter element : ");
        scanf("%d",&a[i]);
    }
    printf("item that needs to be searched\n");
    scanf("%d",&key);
    found=search(a,front,n);
    if(found>=0)
    printf("element found at position %d\n",found);
    else
    printf("element not found\n");
}
int search(int a[],int front, int n)
{
    if(a[front] == key)
    return front;
    else if(front == n)
    return-1;
    else
    return search(a,front+1,n);
}
/*output
student@student:~/22251A3656$ gcc lsrec.c
student@student:~/22251A3656$ ./a.out
enter the number of elements u want to read
5
enter element : 1
enter element : 2
enter element : 3
enter element : 4
enter element : 5
item that needs to be searched
5
element found at position 4
student@student:~/22251A3656$ gcc lsrec.c
student@student:~/22251A3656$ ./a.out
enter the number of elements u want to read
4
```

```
enter element : 23
enter element : 45
enter element : 77
enter element : 44
item that needs to be searched
89
element not found
*/
```

```
/*Write a program to implement binary search using recursion
Created on : 13-6-23
Author: Lasya Nalagandla, 22251A3656
*/
#include<stdio.h>
#include<stdlib.h>
void binarysearch(int ar[],int high,int low,int ele);
void main()
{
	int n,ele;
	printf("Enter number of elements in an array : ");
	scanf("%d",&n);
	int a[n];
	printf("Enter elements: ");
	for(int i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}
	printf("Enter element to search : ");
	scanf("%d",&ele);
	for(int i=0;i<n;i++)
	{
		for(int j=0;j<n-1;j++)
		{
			if(a[j]>a[j+1])
			{
				int temp=a[j];
				a[j]=a[j+1];
				a[j+1]=temp;
			}
		}
	}
	for(int k=0;k<n;k++)
		printf("a[%d] = %d\n",k,a[k]);
	binarysearch(a,n-1,0,ele);
}
void binarysearch(int ar[],int high,int low,int ele)
{
	int mid=(low+high)/2;
	if(low>high)
	{
		printf("Not Found:(\n");
		return;
	}
	else if(ar[mid]==ele)
	{
		printf("Found at position %d!!\n",mid);
		return;
	}
	else
```

```
        {
                if(ar[mid]>ele)
                        binarysearch(ar,mid-1,low,ele);
                else
                        binarysearch(ar,high,mid+1,ele);
        }
}

/*OUTPUTS:
student@user:~/22251a3661$ gcc binaryrec.c
student@user:~/22251a3661$ ./a.out
Enter number of elements in an array : 5
Enter elements: 2
1
54

34
56
Enter element to search : 4
a[0] = 1
a[1] = 2
a[2] = 34
a[3] = 54
a[4] = 56
Not Found:(
student@user:~/22251a3661$ ./a.out
Enter number of elements in an array : 5
Enter elements: 34
678
23
67
12
Enter element to search : 12
a[0] = 12
a[1] = 23
a[2] = 34
a[3] = 67
a[4] = 678
Found at position 0!!
*/
```

```
/*Write a C program to implement selection sort
Created on : 27-6-23
Author: Lasya Nalagandla, 22251A3656
*/
#include<stdio.h>
void selectionsort(int [],int);
void main()
{
    int n,i,a[20];
    printf("Enter no. of elements : ");
    scanf("%d",&n);
    printf("Before sorting the elements are: \n");
    for(i=0;i<n;i++)
    {
        printf("a[%d] : ",i);
        scanf("%d",&a[i]);
    }
    selectionsort(a,n);
    printf("Elements after sorting: \n");
    for(i=0;i<n;i++)
        printf("a[%d] : %d \n",i,a[i]);
}
void selectionsort(int a[],int n)
{
    int pos,i,t,j;
    for(i=0;i<n-1;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
            if(a[j]<a[pos])
            pos=j;
        }
        if(i!=pos)
        {
            t=a[i];
            a[i]=a[pos];
            a[pos]=t;
        }
    }
}
/*output
student@student:~/22251A3656$ gcc selectionsort.c
student@student:~/22251A3656$ ./a.out
Enter no. of elements : 5
Before sorting the elements are:
a[0] : 45
a[1] : 90
a[2] : 87
a[3] : 66
a[4] : 2
```

```
Elements after sorting:
a[0] : 2
a[1] : 45
a[2] : 66
a[3] : 87
a[4] : 90
student@student:~/22251A3656$ gcc selectionsort.c
student@student:~/22251A3656$ ./a.out
Enter no. of elements : 4
Before sorting the elements are:
a[0] : 34
a[1] : 45
a[2] : 69
a[3] : 98
Elements after sorting:
a[0] : 34
a[1] : 45
a[2] : 69
a[3] : 98
student@student:~/22251A3656$ gcc selectionsort.c
student@student:~/22251A3656$ ./a.out
Enter no. of elements : 3
Before sorting the elements are:
a[0] : 98
a[1] : 55
a[2] : 34
Elements after sorting:
a[0] : 34
a[1] : 55
a[2] : 98
*/
```

```c
/*Write a C program to implement insertion sort
Created on : 27-6-23
Author: Lasya Nalagandla, 22251A3656*/

#include<stdio.h>
void insertionsort(int [],int);
void main()
{
    int n,i,a[20];
    printf("Enter no. of elements : ");
    scanf("%d",&n);
    printf("Before sorting the elements are: \n");
    for(i=0;i<n;i++)
    {
        printf("a[%d] : ",i);
        scanf("%d",&a[i]);
    }
    insertionsort(a,n);
    printf("Elements after sorting: \n");
    for(i=0;i<n;i++)
        printf("a[%d] : %d \n",i,a[i]);
}
void insertionsort(int a[],int n)
{
    int p,t,i;
    for(i=1;i<n;i++)
    {
        t=a[i];
        p=i-1;
        while(t<a[p] && p>=0)
        {
            a[p+1]=a[p];
            p=p-1;
        }
        a[p+1]=t;
    }
}
/*output
student@student:~/22251A3656$ gcc insertsort.c
student@student:~/22251A3656$ ./a.out
Enter no. of elements : 5
Before sorting the elements are:
a[0] : 33
a[1] : 98
a[2] : 45
a[3] : 2
a[4] : 12
Elements after sorting:
a[0] : 2
a[1] : 12
a[2] : 33
```

```
a[3] : 45
a[4] : 98
student@student:~/22251A3656$ gcc insertsort.c
student@student:~/22251A3656$ ./a.out
Enter no. of elements : 3
Before sorting the elements are:
a[0] : 32
a[1] : 55
a[2] : 79
Elements after sorting:
a[0] : 32
a[1] : 55
a[2] : 79
student@student:~/22251A3656$ gcc insertsort.c
student@student:~/22251A3656$ ./a.out
Enter no. of elements : 3
Before sorting the elements are:
a[0] : 88
a[1] : 77
a[2] : 44
Elements after sorting:
a[0] : 44
a[1] : 77
a[2] : 88
*/
```

```
/*Write a C program to implement merge sort
Created on : 4-7-23
Author: Lasya Nalagandla, 22251A3656*/

#include<stdio.h>
void msort(int [],int,int);
void merge(int [],int ,int ,int);
void main()
{
    int n,i,a[20];
    printf("Enter no. of elements : ");
    scanf("%d",&n);
    printf("Before sorting the elements are: \n");
    for(i=0;i<n;i++)
    {
        printf("a[%d] : ",i);
        scanf("%d",&a[i]);
    }
    msort(a,0,n-1);
    printf("Elements after sorting: \n");
    for(i=0;i<n;i++)
        printf("a[%d] : %d \n",i,a[i]);
}
void msort(int a[],int left,int right)
{
    int mid;
    if(left<right)
    {
        mid=(left+right)/2;
        msort(a,left,mid);
        msort(a,mid+1,right);
        merge(a,left,mid,right);
    }
}
void merge(int a[],int l,int m,int r)
{
    int temp[10],k=0,j,i;
    for(i=l,j=m+1;i<=m && j<=r; )
    {
        if(a[i]<=a[j])
        {
            temp[k++]=a[i];
            i++;
        }
        else
        {
            temp[k++]=a[j];
            j++;
        }
    }
    while(i<=m)
```

```
        temp[k++]=a[i++];
    while(j<=r)
        temp[k++]=a[j++];
    for(i=0;i<k;i++)
        a[l+i]=temp[i];
}
/*output
student@student:~/22251A3656$ gcc msort.c
student@student:~/22251A3656$ ./a.out
Enter no. of elements : 5
Before sorting the elements are:
a[0] : 2
a[1] : 45
a[2] : 66
a[3] : 73
a[4] : 98
Elements after sorting:
a[0] : 2
a[1] : 45
a[2] : 66
a[3] : 73
a[4] : 98
Enter no. of elements : 5

Before sorting the elements are:
a[0] : 88
a[1] : 77
a[2] : 66
a[3] : 55
a[4] : 44
Elements after sorting:
a[0] : 44
a[1] : 55
a[2] : 66
a[3] : 77
a[4] : 88

Enter no. of elements : 5
Before sorting the elements are:
a[0] : 45
a[1] : 34
a[2] : 77
a[3] : 2
a[4] : 90
Elements after sorting:
a[0] : 2
a[1] : 34
a[2] : 45
a[3] : 77
a[4] : 90
*/
```

```
/*Write a C program to implement quick sort
Created on : 4-7-23
Author: Lasya Nalagandla, 22251A3656*/

#include<stdio.h>
void qsort(int [],int,int);
void swap(int *,int *);
void main()
{
    int n,i,a[20];
    printf("Enter no. of elements : ");
    scanf("%d",&n);
    printf("Before sorting the elements are: \n");
    for(i=0;i<n;i++)
    {
        printf("a[%d] : ",i);
        scanf("%d",&a[i]);
    }
    qsort(a,0,n-1);
    printf("Elements after sorting: \n");
    for(i=0;i<n;i++)
        printf("a[%d] : %d \n",i,a[i]);
}
void qsort(int a[],int left,int right)
{
    int t,l,r,pivot,i;
    l=left;
    r=right;
    if(left<right)
    {
        pivot=a[left];
        while(l<=r)
        {
            while((l<=r)&&a[l]<=pivot) l++;
            while((r>=l)&&a[r]>pivot) r--;
            if(l<r)
            swap(&a[l],&a[r]);
        }
        swap(&a[left],&a[r]);
        qsort(a,left,r-1);
        qsort(a,r+1,right);
    }
}
void swap(int *a,int *b)
{
    int t;
    t=*a;
    *a=*b;
    *b=t;
}
/*
```

```
output
student@student:~/22251A3656$ gcc qsort.c
student@student:~/22251A3656$ ./a.out
Enter no. of elements : 4
Before sorting the elements are:
a[0] : 23
a[1] : 44
a[2] : 3
a[3] : 7
Elements after sorting:
a[0] : 3
a[1] : 7
a[2] : 23
a[3] : 44
student@student:~/22251A3656$ gcc qsort.c
student@student:~/22251A3656$ ./a.out
Enter no. of elements : 4
Before sorting the elements are:
a[0] : 12
a[1] : 13
a[2] : 14
a[3] : 15
Elements after sorting:
a[0] : 12
a[1] : 13
a[2] : 14
a[3] : 15
student@student:~/22251A3656$ gcc qsort.c
student@student:~/22251A3656$ ./a.out
Enter no. of elements : 4
Before sorting the elements are:
a[0] : 98
a[1] : 78
a[2] : 68
a[3] : 48
Elements after sorting:
a[0] : 48
a[1] : 68
a[2] : 78
a[3] : 98
*/
```

```
/*Write a C program to implement heap sort
Created on : 11-7-23
Author: Lasya Nalagandla, 22251A3656*/

#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
int h[50],count=-1;
void insert(int);
int delete();
void swap(int *,int *);
bool isempty();
void main()
{
   int a[10],n,i;
   printf("Before sorting\n");
   printf("Enter number of elements: \n");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("a[%d] : ",i);
      scanf("%d",&a[i]);
   }
   for(i=0;i<n;i++)
      insert(a[i]);
   for(i=n-1;i>=0;i--)
      a[i]=delete();
   printf("after sorting\n");
   for(i=0;i<n;i++)
      printf("%d ",a[i]);
}
void insert(int e)
{
   int i,par;
   h[++count]=e;
   i=count;
   //sift up
   while(i>0)
   {
      par=(i-1)/2;
      if(h[i]>h[par])
      {
         swap(&h[i],&h[par]);
         i=par;
      }
      else
         break;
   }
}
bool isempty()
{
    if(count==-1)
        return true;
    else
        return false;
}
int delete()
{
    int d,par,child;
    if(isempty())
    {
        printf("empty\n");
        exit(0);
    }
    d=h[0];
    h[0]=h[count];//replace root with last element
    count--;
    //sift down
    par=0;child=2*par+1;
    while(child<=count)//if child exists
    {
        //find which child is larger
        if((child<count) && (h[child]<h[child+1]))
            child++;
        if(h[par]<h[child])
        {
            swap(&h[par],&h[child]);
            par=child;
            child=2*par+1;
        }
        else
            break;
     }
     return d;
}
void swap(int *a,int *b)
{
    int t;
    t=*a;
    *a=*b;
    *b=t;
}
/*output
Before sorting
Enter number of elements:  5
a[0] : 23
a[1] : 1
a[2] : 45
a[3] : 7
a[4] : 90
after sorting
1 7 23 45 90 */
```

```
/*Write a program to implement dictionary for the following operations : insert ,delete,search
Created on : 18-7-23
Author: Lasya Nalagandla, 22251A3656
*/
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#define size 11
struct data
{
   int key;
   int value;
   bool nused;
};
void insert(int,int);
int hashfunc(int);
int search(int);
void display();
int delete(int);
struct data ht[size];
int hashfunc(int key)
{
   return (key%size);
};
void main()
{
   int key,value,s,d,ele,e;
   int ch;
   for(int i=0;i<size;i++)
   {
      ht[i].key=-1;
      ht[i].value=-1;
      ht[i].nused=true;
   }
   do
   {
      printf("1.Insert\n2.Delete\n3.Search\n4.Display\n5.Exit\n");
      printf("Enter a choice");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1 : printf("Enter key and value : \n");
                  scanf("%d%d",&key,&value);
                  insert(key,value);
                  break;
         case 2 : printf("Enter the key that needs to be deleted\n");
                  scanf("%d",&ele);
                  d=delete(ele);
                  printf("Deleted key %d is at pos : %d \n",ele,d);
                  break;
         case 3 : printf("Element that needs to be searched ");
```

```
                scanf("%d",&e);
                s=search(e);
                if(s==-1)
                printf("Element not found\n");
                else
                printf("Element found at pos :%d\n",s);
                break;
            case 4 : display();
                break;
            case 5 : exit(0);
                break;
            default : printf("Invalid choice\n");
        }
    }while(1);
}
void insert(int k,int v)
{
    int hashindex,i;
    hashindex=hashfunc(k);
    i=hashindex;
    while(ht[i].key!=-1)
    {
        i=(i+1)%size;
        if(i==hashindex)
        {
            printf("Hashtable is full\n");
            return;
        }
    }
    ht[i].key=k;
    ht[i].value=v;
    ht[i].nused=false;
}
int search(int k)
{
    int i,hashindex;
    hashindex=hashfunc(k);
    i=hashindex;
    while(ht[i].nused!=true)
    {
        if(ht[i].key==k)
            return i;
            i=(i+1)%size;
            if(i==hashindex)
            {
                break;
            }
    }
    return-1;
}
int delete (int k)
```

```
{
    int s;
    s=search(k);
    if(s!=-1)
    {
        ht[s].key=-1;
        ht[s].value=-1;
    }
    return s;
}
void display()
{
    int i;
    for(i=0;i<size;i++)
        printf("key : %d value : %d\n",ht[i].key,ht[i].value);
}
/*output
student@student:~/22251A3656$ gcc dictionary.c
student@student:~/22251A3656$ ./a.out
 test case 1: INSERT
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
23 55
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
89 5
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice4
key :-1 value :-1
key : 23 value : 55
key : 89 value : 5
key :-1 value :-1
key :-1 value :-1
key :-1 value :-1
key :-1 value :-1
key :-1 value :-1
key :-1 value :-1
key :-1 value :-1
key :-1 value :-1
test case 2: insertion-hashtable full
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
2 78
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
7 44
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
34 9
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
87 4
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
6 4
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
77 4
```

```
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
55 9
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
90 8
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
3 88
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
99 46
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
22 6
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice1
Enter key and value :
5 80
Hashtable is full
1.Insert
2.Delete
```

```
3.Search
4.Display
5.Exit
Enter a choice4
key : 77 value : 4
key : 34 value : 9
key : 2 value : 78
key : 55 value : 9
key : 90 value : 8
key : 3 value : 88
key : 6 value : 4
key : 7 value : 44
key : 99 value : 46
key : 22 value : 6
key : 87 value : 4
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice
test case 3: DELETE
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice2
Enter the key that needs to be deleted
89
Deleted key 89 is at pos : 2
test case 4 : SEARCH
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice3
Element that needs to be searched 23
Element found at pos :1
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice3
Element that needs to be searched 89
Element not found
test case 5: delete and search on empty hashtable
1.Insert
2.Delete
```

```
3.Search
4.Display
5.Exit
Enter a choice2
Enter the key that needs to be deleted
23
Deleted key 23 is at pos :-1
1.Insert
2.Delete
3.Search
4.Display
5.Exit
Enter a choice3
Element that needs to be searched 23
Element not found
1.Insert
2.Delete
3.Search
4.Display
5.Exit

*/
```

```
/* Write a program to implement depth first search traversal of the graph
Created on : 22-7-23
Author: Lasya Nalagandla, 22251A3656*/

#include<stdio.h>
int top=-1;
int stack[20];
void push(int x);
int pop();
void push(int x)
{
   top=top+1;
   stack[top]=x;
}
int pop()
{
   int d=stack[top];
   top=top-1;
   return d;
}
void dfs(int adj[20][20],int visited[],int start,int n);
void dfs(int adj[20][20],int visited[20],int start,int n)
{
   printf("%d \n",start);
   visited[start]=1;
   for(int i=0;i<n;i++)
   {
      if(adj[start][i]==1&&visited[i]!=1)
         push(i);
   }
   start=pop();
   while(top!=-1)
   {
      if(visited[start]!=1)
      dfs(adj,visited,start,n);
   }
}
void main ()
{
  int visited[20] = { 0 }, adj[20][20]={0}, i, j, v1, v2, type, start,n;
  printf ("Enter no of vertices:");
  scanf ("%d", &n);
  printf ("Enter directed(0) or undirected(1)");
  scanf ("%d", &type);
  do
   {
      printf ("enter the edge(source to desti.to stop-1-1)");
      scanf ("%d%d", &v1, &v2);
```

```
    if (v1 ==-1 && v2 ==-1)
         break;
    else
       {
        adj[v1][v2] = 1;
        if (type)
         adj[v2][v1] = 1;
       }
  } while (1);
  printf("adj matrix is\n");
   for(i=0;i<n;i++)
   {
    for(j=0;j<n;j++)
      printf("%d ",adj[i][j]);
    printf("\n");
   }
  printf("Enter the starting vertex:");
  scanf("%d",&start);
  dfs(adj,visited,start,n);
}

/*OUTPUTS :
Enter no of vertices:8
Enter directed(0) or undirected(1)1
enter the edge(source to desti.to stop-1-1)0 1
enter the edge(source to desti.to stop-1-1)0 2
enter the edge(source to desti.to stop-1-1)1 3
enter the edge(source to desti.to stop-1-1)1 4
enter the edge(source to desti.to stop-1-1)2 5
enter the edge(source to desti.to stop-1-1)2 6
enter the edge(source to desti.to stop-1-1)3 7
enter the edge(source to desti.to stop-1-1)4 7
enter the edge(source to desti.to stop-1-1)5 7
enter the edge(source to desti.to stop-1-1)6 7
enter the edge(source to desti.to stop-1-1)-1-1
adj matrix is
0 1 1 0 0 0 0 0
1 0 0 1 1 0 0 0
1 0 0 0 0 1 1 0
0 1 0 0 0 0 0 1
0 1 0 0 0 0 0 1
0 0 1 0 0 0 0 1
0 0 1 0 0 0 0 1
0 0 0 1 1 1 1 0
Enter the starting vertex:0
0
2
6
```

```
7
5
4
1
3
*/
```

```
/* Write a program to implement breadth first search traversal of the graph
Created on : 22-7-23
Author: Lasya Nalagandla, 22251A3656*/
*/

#include <stdio.h>
int q[20];
int rear=-1,front=-1,n;
void insert(int ele)
{
    q[++rear]=ele;
    if (front==-1)
        front=0;
}
int del()
{
    if(front==-1 || front>rear)
        return(-1);
    else
        return(q[front++]);
}
void bfs(int adj[][20],int visited[],int start);
void bfs(int adj[][20],int visited[],int start)
{
    int i;
    insert(start);
    visited[start]=1;
    while(front<=rear)
    {
        start = del();
        printf("%d\n",start);
        for(i=0;i<n;i++)
        {
            if(adj[start][i] && !visited[i])
            {
                insert(i);
                visited[i]=1;
            }
        }
    }
}
void main ()
{
  int visited[20] = { 0 }, adj[20][20]={0}, i, j, v1, v2, type, start;
  printf ("Enter no of vertices:");
  scanf ("%d", &n);
  printf ("Enter directed(0) or undirected(1)");
  scanf ("%d", &type);
```

```
 do
  {
    printf ("enter the edge(source to desti.to stop-1-1)");
    scanf ("%d%d", &v1, &v2);
    if (v1 ==-1 && v2 ==-1)
         break;
    else
       {
        adj[v1][v2] = 1;
        if (type)
         adj[v2][v1] = 1;
       }
  } while (1);
  printf("adj matrix is\n");
   for(i=0;i<n;i++)
   {
    for(j=0;j<n;j++)
      printf("%d ",adj[i][j]);
    printf("\n");
   }
  printf("Enter the starting vertex:");
  scanf("%d",&start);
  bfs(adj,visited,start);
}

/*OUTPUTS:
Enter no of vertices:3
Enter directed(0) or undirected(1)
1
enter the edge(source to desti.to stop-1-1)0 1
enter the edge(source to desti.to stop-1-1)0 2
enter the edge(source to desti.to stop-1-1)1 2
enter the edge(source to desti.to stop-1-1)-1-1
adj matrix is
0 1 1
1 0 1
1 1 0
Enter the starting vertex:0
0
1
2

Enter no of vertices:10
Enter directed(0) or undirected(1)0
enter the edge(source to desti.to stop-1-1)0 1
enter the edge(source to desti.to stop-1-1)0 2
enter the edge(source to desti.to stop-1-1)0 3
enter the edge(source to desti.to stop-1-1)
```

```
1 4
enter the edge(source to desti.to stop-1-1)2 4
enter the edge(source to desti.to stop-1-1)3 2
enter the edge(source to desti.to stop-1-1)3 5
enter the edge(source to desti.to stop-1-1)3 6
enter the edge(source to desti.to stop-1-1)4 7
enter the edge(source to desti.to stop-1-1)5 2
enter the edge(source to desti.to stop-1-1)5 7
enter the edge(source to desti.to stop-1-1)6 7
enter the edge(source to desti.to stop-1-1)6 8
enter the edge(source to desti.to stop-1-1)9 8
enter the edge(source to desti.to stop-1-1)9 7
enter the edge(source to desti.to stop-1-1)-1-1
adj matrix is
0 1 1 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 1 0 0 1 1 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0
Enter the starting vertex:0
0
1
2
3
4
5
6
7
8
*/
```

```
/*Write a program to implement binary search tree insertion, search and traverse.
Created on : 25-7-23
Author: Lasya Nalagandla, 22251A3656
*/

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
typedef struct node
{
 int data;
 struct node *left,*right;
}bstnode;
bstnode* stack[20];
int top=-1,n;
void push(bstnode* r)
{
              stack[++top]=r;
}
bstnode* pop()
{
 if(top==-1)
              return NULL;
   else
              return(stack[top--]);
}
 bstnode * insert(bstnode*, int);
void preorder(bstnode* );
void inorder(bstnode* );
void postorder(bstnode* );
void inorder_iterative(bstnode*);
int rsearch(bstnode*,int);
bool isempty(bstnode*);
void main()
{
 int num,ch,data,s,d;
 bstnode *root=NULL;

 while(1)
 {
  printf("\n1.Insert 2.Preorder 3.Inorder 4.Postorder 5.search 6.inorderr-iter 7.exit");
  printf("\nEnter your choice: ");scanf("%d",&ch);
  switch(ch)
  {
   case 1: printf("Enter integer: ");
                 scanf("%d",&num);
                 root=insert(root,num);
                 break;
   case 2: if(isempty(root))
            printf("Tree is empty");
                     else
                     {
                      printf("Preoder:");
                      preorder(root);
                     }
                    break;
   case 3:if(isempty(root))
            printf("Tree is empty");
                     else
                     {
                      printf("inorder:");
                      inorder(root);
                     }
                     break;
   case 4:if(isempty(root))
            printf("Tree is empty");
                     else
                     {
                      printf("Postorder:");
                      postorder(root);
                     }
                                 break;
   case 5:  printf("Enter the element to be searched:");
                      scanf("%d",&data);
                      s=rsearch(root,data);
                      if(s==1) printf("found"); else printf("not found");
                    break;
   case 6: if(isempty(root))
            printf("Tree is empty");
                     else
                     {
                      printf("inorder(iterative):");
                      inorder_iterative(root);
                     }
                     break;
   case 7: exit(0);
   default: printf("Wrong choice....\n");
  }
 }
}
bool isempty(bstnode *root)
{
   if (root==NULL)
     return true;
   else
     return false;
}
bstnode * insert(bstnode *root, int n)
{
  bstnode *newn;
 if(root==NULL)
 {
  newn=(bstnode*) malloc (sizeof(bstnode));
  newn->left=newn->right=NULL;
  newn->data=n;
  return(newn);
 }
 if(n<root->data)
  root->left=insert(root->left,n);
 else
  root->right=insert(root->right,n);
 return(root);
}

void preorder(bstnode *r)
{

 if(r!=NULL)
 {
  printf("%d ",r->data);
  preorder(r->left);
  preorder(r->right);
 }
}
void inorder(bstnode *r)
{
 if(r!=NULL)
 {
  inorder(r->left);
  printf("%d ",r->data);
  inorder(r->right);
 }
}
void postorder(bstnode *r)
{
 if(r!=NULL)
 {
  postorder(r->left);
  postorder(r->right);
  printf("%d ",r->data);
 }
}
void inorder_iterative(bstnode *r)
{
  while(r!=NULL)
 {
    push(r);
    if(r->left!=NULL)
        r=r->left;
    else
    {
        do
        {
            if(top==-1)
               break;
            r=pop();
            printf("%d ",r->data);
        }while(r->right==NULL);
        r=r->right;
    }
 }
}
int rsearch(bstnode *r,int n)
{
   if(isempty(r)) return 0;
   else if(n==r->data)
     return 1;
   if(n<r->data)
     return(rsearch(r->left,n));
   else
     return(rsearch(r->right,n));
}

/*Output

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.search 6.inorderr-iter 7.exit
Enter your choice: 1
Enter integer: 22

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.search 6.inorderr-iter 7.exit
Enter your choice: 1
Enter integer: 65

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.search 6.inorderr-iter 7.exit
Enter your choice: 1
Enter integer: 88

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.search 6.inorderr-iter 7.exit
Enter your choice: 1
Enter integer: 54

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.search 6.inorderr-iter 7.exit
Enter your choice: 1
Enter integer: 90

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.search 6.inorderr-iter 7.exit
Enter your choice: 1
Enter integer: 7

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.search 6.inorderr-iter 7.exit
Enter your choice: 2
Preoder:22 7 65 54 88 90

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.search 6.inorderr-iter 7.exit
```

```
Enter your choice: 3
inorder:7 22 54 65 88 90

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.search 6.inorderr-iter 7.exit
Enter your choice: 4
Postorder:7 54 90 88 65 22

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.search 6.inorderr-iter 7.exit
Enter your choice: 5
Enter the element to be searched:6
not found

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.search 6.inorderr-iter 7.exit
Enter your choice: 5
Enter the element to be searched:88
Found

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.search 6.inorderr-iter 7.exit
Enter your choice: 6
inorder(iterative):7 22 54 65 88 90
*/
```